УДК 003.26
DOI: 10.25559/SITITO.14.201802.419-425

ОБ УНИВЕРСАЛЬНОМ ДРЕВОВИДНОМ РЕЖИМЕ ВЫРАБОТКИ ХЭШ-КОДА

Д.С. Богданов[1], Ф.А. Дали[2], В.О. Миронкин[3]
[1] Лаборатория ТВП, г. Москва, Россия
[2] Технический комитет по стандартизации «Криптографическая защита информации», г. Москва, Россия
[3] Национальный исследовательский университет «Высшая школа экономики», г. Москва, Россия

ON THE UNIVERSAL TREE MODE OF HASH CODE GENERATION

Dmitriy S. Bogdanov[1], Farkhad A. Dali[2], Vladimir O. Mironkin[3]
[1] Laboratory of TVP, Moscow, Russia
[2] Technical Committee of Standardization Cryptography Information Security, Moscow, Russia
[3] National Research University Higher School of Economics, Moscow, Russia

Ключевые слова

Хэш-функция; хэш-код; распараллеливание вычислений; дерево хэширования; деревья Меркла; режим; алгоритм; форматирование.

Аннотация

Классические подходы к построению режимов работы хэш-функций, основанные на использовании итеративных процедур, не позволяют обеспечить эффективную обработку больших объемов данных и не могут быть адаптированы к параллельным вычислительным архитектурам. Это касается как российского криптографического стандарта ГОСТ Р 34.11-2012, определяющего алгоритм и процедуру вычисления хэш-функции, так и многих других зарубежных стандартов (например, SHA-3). Отсутствие действующих стандартов в части режимов работы хэш-функций ГОСТ Р 34.11-2012 создает острую необходимость разработки отечественного стандарта параллелизуемого режима выработки хэш-кода.

Настоящая статья посвящена исследованию и разработке новых режимов выработки хэш-кода, допускающих эффективное распараллеливание процесса вычислений и обеспечивающих криптографическую стойкость, удовлетворяющую современным требованиям. Данная работа продолжает исследования, проводимые авторами, и предлагает принципиально новый универсальный древовидный режим выработки хэш-кода («FT-режим»), построенный на основе l-арных деревьев хэширования и позволяющий применять в качестве механизма формирования узлов дерева любое сжимающее отображение. При этом стойкость режима полностью определяется стойкостью соответствующего сжимающего отображения. Так, в частности, для формирования узлов дерева хэширования, наряду с функциями сжатия и хэш-функциями, FT-режим допускает использование блочных шифров, подстановочных преобразований и т.д. В дополнение к этому FT-режим исключает основные функциональные недостатки известных древовидных режимов выработки хэш-кода, влияющие на их эксплуатационно-технические и криптографические качества.

В рамках настоящих исследований вычислен ряд характеристик FT-режима, а также проведен сравнительный анализ временной и вычислительной трудоемкостей реализаций FT-режима и некоторых иностранных режимов древовидного хэширования, по результатам которого разработанный режим не уступает ни одному из рассмотренных.

Об авторах:

Богданов Дмитрий Сергеевич, старший научный сотрудник, Лаборатория ТВП (117418, Россия, г. Москва, Нахимовский проспект, д. 47), ORCID: http://orcid.org/0000-0001-6178-6420, bogdanovds@rambler.ru

Дали Фархад Алишерович, эксперт, Технический комитет по стандартизации «Криптографическая защита информации» (ТК 26) (127287, Россия, г. Москва, Старый Петровско-Разумовский проезд, д. 1/23, стр. 1), ORCID: http://orcid.org/0000-0002-3344-9766, dali_fa@tc26.ru

Миронкин Владимир Олегович, старший преподаватель, кафедра компьютерная безопасность, Национальный исследовательский университет «Высшая школа экономики» (123458, Россия, г. Москва, ул. Таллинская, д. 34), ORCID: http://orcid.org/0000-0002-5606-7509, mironkin.v@mail.ru

Keywords

Hash function; hash code; parallelization of calculations; hash tree; Merkle trees; mode; algorithm; formatting.

Abstract

Classical approaches to the construction of hash function modes, based on the using of iterative procedures, do not allow efficient processing of large amounts of data and can't be adapted to parallel computing architectures. It applies to both the Russian cryptographic standard GOST R 34.11-2012, which determines the algorithm and procedure for calculating the hash function, as well as many other foreign standards (for example, SHA-3). The absence of standards for parallelized modes for the hash functions of GOST R 34.11-2012 creates an urgent need for the development of the domestic standard of the parallelized mode of hash code.

This article is devoted to the research and development of new modes of hash code generation that allow efficient parallelization of the computation process and provide cryptographic resistance satisfying modern requirements. This work continues the research carried out by the authors, and offers a fundamentally new tree mode of hash code generation ("FT-mode"), based on l-ary hash trees and allowed to use any compression mapping for a mechanism of forming tree nodes. The resistance of the mode is completely determined by the resistance of the corresponding compressive mapping. In particular, the FT-mode allows using block ciphers and substitution transformations to form nodes of a hash tree along with compression functions and hash functions. In addition, the FT-mode excludes the main functional disadvantages of the known tree modes of hash code generation that affect their operational, technical and cryptographic quality.

Within the framework of the present research a number of characteristics of FT-mode are calculated, and a comparative analysis of the time and computational complexity of implementations of FT-mode and some foreign tree hash modes is carried out. The corresponding results showed that the developed mode is not inferior to any of the considered modes.

1. Введение

Одним из основных требований, предъявляемых к современным итеративным методам хэширования, предназначенным для решения задач обеспечения целостности информации и аутентичности субъектов (объектов) информационного взаимодействия, является возможность их адаптации к параллельным вычислительным архитектурам. Далеко не все стандартизированные решения в области синтеза хэш-функций имеют высокую степень распараллеливания (это касается и отечественного стандарта ГОСТ Р 34.11-2012 [1]).

Разработка режимов распараллеливания работы хэш-функций, удовлетворяющих полному спектру современных эксплуатационно-технических и криптографических требований, становится все более актуальной в силу их применения в различных практических приложениях (проверка цифровой подписи, передача больших объемов данных и т.д.). Так наиболее ярким примером их использования в повседневной жизни является хорошо известная технология Blockchain [2, 3, 4], на основе которой работают современные платежные системы Bitcoin и Litecoin, а также формируются различные финансовые Blockchain-платформы.

Существующие режимы распараллеливания работы хэш-функций [5, 6, 7, 8, 9], описанные в [10], обладают рядом существенных функциональных недостатков, которые в ряде случаев отрицательно сказываются как на их эффективности, так и на криптографической стойкости.

В настоящей статье описан новый универсальный древовидный режим выработки хэш-кода – «FT-режим» (от английского «**F**ormat **T**ree's mode»), исключающий соответствующие недостатки и позволяющий использовать любой криптографический примитив (функции сжатия, хэш-функции и т.д.). При этом стойкость режима полностью определяется стойкостью соответствующего сжимающего отображения.

2. Описание FT-режима

При определении FT-режима будем использовать обозначения и терминологию, введенные в работах [11, 12]. Так, в частности,

1. n – длина исходного сообщения $M \in V^*$;
2. m – размерность области определения внутренней функции h;
3. r – длина векторов, используемых для записи служебной информации;
4. t – размерность области значений внутренней функции h;
5. $LSB_n : V^* \to V_n$ – отображение, ставящее в соответствие вектору $z_{k-1} \| \ldots \| z_1 \| z_0$, $k \geq n$, вектор $z_{n-1} \| \ldots \| z_1 \| z_0$;
6. $MSB_n : V^* \to V_n$ – отображение, ставящее в соответствие вектору $z_{k-1} \| \ldots \| z_1 \| z_0$, $k \geq n$, вектор $z_{k-1} \| z_{k-2} \| \ldots \| z_{k-n}$;
7. $\boxplus$ - операция сложения в кольце $\mathbb{Z}_{2^r}$;
8. $\overline{x}_r$ - представление числа x в виде двоичного вектора длины r.

Древовидный FT-режим выработки хэш-кода основан на *l*-арных деревьях хэширования [11], являющихся обобщением деревьев Меркла [13], в которых в качестве механизма формирования узлов используется произвольное отображение $h : V_m \to V_t$, $m > t$ (далее –- внутренняя функция). При этом внутренняя функция h представляется в следующем виде:

$$h: V_{r+tl} \to V_t, \qquad (1)$$

где $l > 1$ – арность дерева хэширования, tl – число бит, выделенных под блоки сообщения, а $2^r \geq l^3 (l-1)^{-1} \left\lceil \frac{n+1}{tl} \right\rceil$.

Замечание 1. *Величина r определяет размер векторов, предназначенных для записи элементов счетчиковой последовательности, используемой при нумерации узлов дерева хэширования, а также максимальное значение длины обрабатываемых сообщений. Так, при фиксированных t, l, r FT-режим позволяет вычислять хэш-код от сообщений длины* $n \leq \frac{2^r t(l-1)}{l^2} - tl - 1$.

Использование нумерации узлов дерева хэширования в процессе выработки хэш-кода исключает возможность построения второго прообраза за счет появления внутренних коллизий [14, 15]. Действительно, если исходное сообщение M подобрано так, что, оно не требует форматирования, то при появлении внутренней коллизии (совпадении значений узлов дерева хэширования) второй прообраз может быть построен за счет замены поддеревьев с корнями в соответствующих узлах (рис. 1).

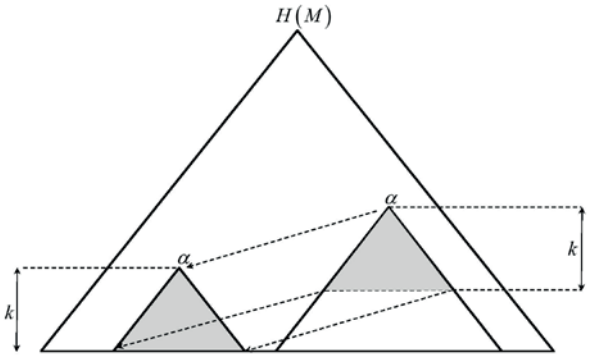

Рис. 1. Построение второго прообраза за счет внутренней коллизии

Fig. 1. Construction of the second prototype due to internal collision

В свою очередь, при нумерации всех узлов дерева хэширования вероятность появления внутренней коллизии становится равной нулю, что исключает появление указанных поддеревьев.

Перейдем к описанию режима выработки хэш-кода. FT-режим состоит из следующих процедур:

1. Дополнение исходного сообщения M служебной информацией;
2. Форматирование сообщения (приведение сообщения к требуемой длине);
3. Построение полного дерева.

Процедура преобразования слоев дерева хэширования является итерационной. Каждая итерация представляет собой обработку текущего слоя дерева с использованием функции сжатия $g_{ml^p} : \mathbb{Z}_{2^r} \times V_{ml^p} \to V_{ml^{p-1}}$, $p \in \mathbb{N} \cup \{0\}$:

$$g_{ml^p}\left(x, A_1 \| A_1 \ldots \| A_{l^p}\right) = h(A_1) \| \ldots \| h(A_i) \| \overline{x \boxplus 1_r} \| \ldots \| h\left(A_{l^{p-1}+1}\right) \ldots \| h\left(A_{l^p}\right) \| \overline{x \boxplus l^{p-1}}_r,$$

где $A_i \in V_m$, $i = \overline{1, l^p}$, $x \in \mathbb{Z}_{2^r}$.

Таким образом, алгоритм выработки хэш-кода $H(M)$ произвольного сообщения $M \in V_n$ на основе функции g_{ml^p} имеет вид:

Алгоритм 1

1. Процедура дополнения;

1.1. Сообщение M длины n дополняем битом 1 и делим полученное сообщение на блоки длины $m-r$:

$$M \| 1 = M^{(1)} \| \ldots \| M^{(p)},$$

где $p = \left\lceil \frac{n+1}{m-r} \right\rceil$;

1.2. Если длина блока $M^{(p)}$ не превосходит $m-r$, дополняем его вектором вида $(00\ldots0)$ до длины $m-r$;

1.3. Каждый блок дополняем r-битным представлением его порядкового номера, формируя сообщение M' длины $n' = m\left\lceil \frac{n+1}{m-r} \right\rceil$:

$$M' = M^{(1)} \| \bar{1}_r \| \ldots \| M^{(p)} \| (00\ldots0) \| \bar{p}_r;$$

2. Процедура форматирования;

2.1. Вычисляем максимальное число $\tau \in \mathbb{N} \cup \{0\}$ такое, что $n' \geq ml^\tau$;

2.2. Формируем вектор M_1:

- если $n' = ml^\tau$, то $M_1 = M'$;
- если $n' > ml^\tau$, подвектор $MSB_{n'-ml^\tau}(M')$ дополняем вектором $\bar{a}$ длины $m(l-1)-c$:

$$\bar{a} = (\underbrace{0,\ldots,0}_{m-r} \| \overline{p \boxplus 1}_r \| \underbrace{0,\ldots,0}_{m-r} \| \overline{p \boxplus 2}_r \| \ldots \| \underbrace{0,\ldots,0}_{m-r} \| \overline{p \boxplus s}_r),$$

где $s = l - 1 - \frac{c}{m}$, c – остаток от деления $n' - ml^\tau$ на $m(l-1)$. При этом полагаем $p = p + s$ и

$$M_1 = g_{l\Delta'}\left(p, \bar{a} \| MSB_{n'-ml^\tau+\Delta'}(M')\right) \| LSB_{ml^\tau-\Delta'}(M'),$$

где $\Delta' = \frac{n' - ml^\tau + m(l-1) - c}{l-1}$;

3. Процедура построения дерева;

4.

4.1. Полагаем $i = 1$, $j = p$;

4.2. Если $i < \tau + 1$, вычисляем $M_{i+1} = g_{ml^{\tau-i+1}}(j, M_i)$, полагаем $i = i+1$, $j = j + \frac{p}{l^i}$ и переходим к шагу 3.2, в противном случае полагаем $H(M) = MSB_t(M_i)$, и алгоритм заканчивает работу.

Замечание 2.

В отличие от известных режимов (например, Sakura [6]) дополнение вектора M' на этапе форматирования осуществляется до применения внутренней функции h, что исключает возможность проведения атак, основанных на дополнениях верхних слоев деревьев хэширования. При этом максимальная длина вектора специального вида, используемого для дополнения, не превосходит величины $m(l-1)-1$, что гораздо эффективнее, чем, например, в режиме MD6 [9].

Следует также отметить, что использование форматирования до построения первого слоя дерева хэширования обеспечивает равномерность использования блоков сообщения, так как каждый блок вносит один и тот же «вклад» (значение длины пути от узла, соответствующего указанному блоку, до корня дерева хэширования) при формировании хэш-кода за исключением, быть может, вектора $\overline{a} \,\|\, MSB_{n'-ml^{\tau}+\Delta'}(M')$ («вклад» бит указанного вектора может быть ровно на единицу больше «вклада» остальных бит сообщения M).

На рис. 2 схематично изображена процедура форматирования сообщения до длины, гарантирующей на следующем этапе построение полного дерева хэширования.

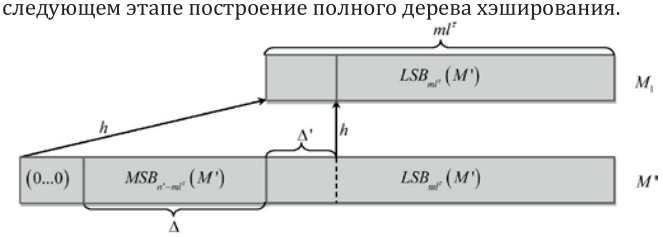

Рис. 2. Форматирование дополненного сообщения M'
Fig. 2. Formatting the amended message M'

Определение 1. *Под вычислительной трудоемкостью T режима выработки хэш-кода будем понимать общее число элементарных операций (количество вычислений значений $h(x), x \in V_m$), необходимых для выработки хэш-кода.*

Определение 2. *Под временной трудоемкостью T^* режима выработки хэш-кода будем понимать общее время, затраченное на выработку хэш-кода.*

Теорема 1. *Пусть $h: V_m \to V_t$ и на вход алгоритма 1 подается сообщение M произвольной длины $n \in \mathbb{N}$. Тогда при $m \geq 2t$*

- если $\lceil \frac{n+1}{m-r} \rceil = l^{\tau}$, то $T = \frac{l^{\tau+1}-1}{l-1}$;
- если $\lceil \frac{n+1}{m-r} \rceil > l^{\tau}$, то

$$T = l\frac{\left\lceil \frac{n+1}{m-r} \right\rceil - l^{\tau} + (l-1) - c}{l-1} + \frac{l^{\tau+1}-1}{l-1},$$

где $m = tl + r$, c – остаток от деления $\left\lceil \frac{n+1}{m-r} \right\rceil - l^{\tau}$ на $l-1$

.

Замечание 3.

Общий объем памяти, выделяемой для выработки хэш-кода сообщения длины n бит при использовании внутренней функции h с параметрами t, l, r, не превосходит $m \cdot \left(\left\lceil \frac{n+1}{tl} \right\rceil + l - 1 \right)$ бит.

В таблице 1 представлены значения вычислительной трудоемкости некоторых древовидных режимов выработки хэш-кода (при оптимальных значениях высот деревьев хэширования), включая FT-режим, для сообщений с заданным числом блоков и значением параметра арности, равным 2 (случай использования деревьев Меркла). При этом полученные значения не зависят от выбора внутренней функции h, поскольку измерения выполнены в элементарных операциях. При этом отметим, что в таблице 1 отсутствуют данные по режиму MD6, так как его параметр арности фиксирован и равен 4.

Таблица 1. Значения T для некоторых древовидных режимов при $l = 2$

Table 1. Values of T for some tree-like modes at $l = 2$

Количество блоков	Sakura	Blake/Phash/Skein	FT-режим
$2^7 - 1$	253	254	253
2^7	255	255	255
$2^7 + 1$	257	264	257
$2^{12} - 1$	8189	8190	8189
2^{12}	8191	8191	8191
$2^{12} + 1$	8193	8205	8193
$2^{17} - 1$	262141	262142	262141
2^{17}	262143	262143	262143
$2^{17} + 1$	262145	262162	262145
$2^{22} - 1$	8388605	8388606	8388605
2^{22}	8388607	8388607	8388607
$2^{22} + 1$	8388609	8388631	8388609
$2^{27} - 1$	268435453	268435454	268435453
2^{27}	268435455	268435455	268435455
$2^{27} + 1$	268435457	268435484	268435457

В таблице 2 представлены результаты аналогичных измерений в случае, когда параметр арности $l = 4$. При этом в таблице отсутствует режим Sakura, так как он использует фиксированный параметр арности равный 2.

Таблица 2. Значения T для некоторых древовидных режимов при $l = 4$

Table 1. Values of T for some tree-like modes at $l = 4$

Количество блоков	MD6	Blake/Phash/Skein	FT-режим
$4^7 - 1$	21845	21844	21845
4^7	21845	21845	21845
$4^7 + 1$	21877	21854	21849
$4^{12} - 1$	22369621	22369620	22369621
4^{12}	22369621	22369621	22369621
$4^{12} + 1$	22369673	22369635	22369625
$4^{17} - 1$	22906492245	22906492244	22906492245
4^{17}	22906492245	22906492245	22906492245
$4^{17} + 1$	22906492286	22906492264	22906492249

Следствие 1. *При условии теоремы 1 справедливо неравенство* $T^* < \sum_{i=0}^{\tau+1} \left\lceil \frac{l^{\tau-i+1}}{S} \right\rceil$, *где* S *- количество реализуемых параллельных процессов вычислений.*

В таблице 3 приведено время работы FT-режима для различных значений длины сообщения и параметра арности l (при использовании функции Keccak [16] в качестве внутренней функции h).

Таблица 3. Значения T^* для FT-режима

Table 3. Values T^* for the FT mode

Длина сообщения, байты	Параметр арности	Время, сек.
2^{14}	2	0,035
2^{20}	2	1,563
2^{22}	2	6,117
2^{22}	22	2,957
2^{23}	23	5,239
2^{24}	2	23,502
2^{24}	24	11,833
2^{25}	2	31,686
2^{25}	25	17,095

Замечание 4.

Все расчеты в статье проведены с помощью вычислительной системы со следующими параметрами: процессор Intel Core i3 7100U 2400 МГц (2 ядра), видеокарта NVIDIA GeForce 940MX (2 Гб), операционная система Windows 10 (Home), оперативная память 4 Гб DDR4. Программная часть написана на PyCharm (Community Edition) 2017.2.4, при этом используемое количество параллельных процессов вычислений $S = 4$.

На рис. 3 изображена зависимость T^* от длины сообщения при фиксированном значении параметра арности $l = 2$ (график А) и при оптимальном выборе значений параметра арности, зависящем от длины исходного сообщения (график Б).

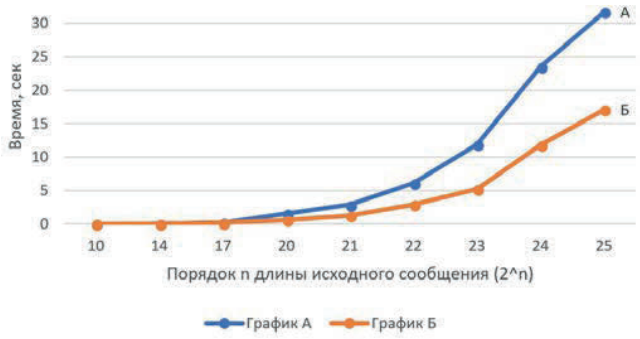

Рис. 3. Зависимость T^* от длины сообщения в FT-режиме

Fig. 3. Dependence of T^* on message length in FT mode

Отметим еще одно преимущество FT-режима – возможность с логарифмической вычислительной сложностью от длины исходного сообщения (в отличие, например, от режима Phash [8]) выполнять проверку принадлежности блоков исходному сообщению и пересчитывать значение хэш-кода при изменении произвольного блока. Это обеспечивается за счет того, что в формируемых деревьях хэширования из каждого листа выходит и при том единственный путь в корень дерева. Например, при изменении одного отдельного узла первого слоя (после форматирования) необходимо пересчитывать только значения в узлах, лежащих на пути из этого узла в корень – всего таких узлов $k-1$, где $k : n_1 = ml^k$ (либо k, если изменение произошло в форматируемой части сообщения), а при изменении значений в $p > 1$ блоках исходного сообщения M необходимо пересчитывать не более pk значений (т.к. возможно пересечение соответствующих путей).

3. Заключение

Описанный универсальный древовидный режим выработки хэш-кода («FT-режим»), построенный на основе *l*-арных деревьев и допускающий эффективное распараллеливание вычислений, позволяет использовать любое сжимающее отображение для формирования узлов дерева хэширования (в том числе и хэш-функцию «Стрибог»), полностью определяющее стойкость режима в целом. Для данного режима вычислены некоторые численные характеристики и описан ряд свойств, влияющих на его криптографическую стойкость. Проведены численные эксперименты и сравнительный анализ с существующими режимами [10].

Список использованных источников

[1] *Kazymyrov A., Kazymyrova V.* Algebraic Aspects of the Russian Hash Standard GOST R 34.11-2012, 2013. 18 p. URL: http://eprint.iacr.org/2013/556.pdf (дата обращения: 11.04.2018).

[2] *Dourado E., Brito J.* Cryptocurrency // The New Palgrave Dictionary of Economics, Online Edition, 2014. 10 p. URL: https://www.mercatus.org/system/files/cryptocurrency-article.pdf (дата обращения: 11.04.2018).

[3] *Решевский М.* Золотая лихорадка XXI век [Электронный ресурс] // Computer Bild. 2011, № 17. С. 64-69.
[4] *Ammous S.* The Bitcoin Standard: The Decentralized Alternative to Central Banking. New York: John Wiley and Sons Inc, 2018. Pp. 304.
[5] *Aumasson J.-P., Neves S., Wilcox-O'Hearn Z., Winnerlein C.* BLAKE2 – fast secure hashing, 2013. URL: https://blake2.net (дата обращения: 11.04.2018).
[6] *Bertoni G., Daemen J., Peeters M., Van Assche G.* Sakura: a flexible coding for tree hashing [Электронный ресурс] // Cryptology ePrint Archive. Report 2013/231, 2013. URL: http://eprint.iacr.org (дата обращения: 11.04.2018).
[7] *Ferguson N., Lucks S., Schneier B., Whiting D., Bellare M., Kohno T., Callas J.* The Skein Hash Function Family // Schneier on Security, Version 1.3. – 1 October, 2010. 92 p. URL: https://www.schneier.com/academic/skein (дата обращения: 11.04.2018).
[8] *Kaminsky A., Radziszowski S.P.* A case for a parallelizable hash // Proceedings of 2008 IEEE Military Communications Conference MILCOM 2008, San Diego, CA, 2008. Pp. 1-7. DOI: 10.1109/MILCOM.2008.4753182
[9] *Rivest R.L., Agre B., Bailey D.V., Crutchfield C., Dodis Y., Fleming K.E., Khan A., Krishnamurthy J., Lin Y., Reyzin L., Shen E., Sukha J., Sutherland D., Tromer E., Yin Y.L.* The MD6 hash function – a proposal to NIST for SHA-3, Submission to NIST, October, 2008. URL: http://groups.csail.mit.edu/cis/md6 (дата обращения: 11.04.2018).
[10] *Дали Ф.А., Миронкин В.О.* Обзор подходов к построению древовидных режимов работы некоторых хэш-функций // Проблемы информационной безопасности. Компьютерные системы. 2017. № 2. С. 46-55.
[11] *Дали Ф.А., Миронкин В.О.* О некоторых моделях древовидного хэширования // Обозрение прикладной промышленной математики. 2017. Т. 24, №. 4. С. 241-244.
[12] *Дали Ф.А., Миронкин В.О.* О древовидных режимах работы хэш-функций // Проблемы информационной безопасности. Компьютерные системы. 2018. № 1. С. 113-121.
[13] *Merkle R.C.* Secrecy, authentication, and public key systems. PhD thesis, UMI Research Press, 1982. 104 p.
[14] *Дали Ф.А., Миронкин В.О.* О вероятностных характеристиках одного класса моделей древовидного хэширования // Обозрение прикладной промышленной математики. 2016. Т. 23, №. 4. С. 345-347.
[15] *Миронкин В.О., Урусов М.И.* О коллизиях деревьев Меркла // Обозрение прикладной промышленной математики. 2018. Т. 25, № 1. С. 84-88.
[16] Bertoni G., Daemen J., Peeters M., Van Assche G. Keccak / Johansson T., Nguyen P.Q. (eds) // Advances in Cryptology – Eurocrypt 2013. Eurocrypt 2013. Lecture Notes in Computer Science. Vol. 7881. Springer, Berlin, Heidelberg, 2013. DOI: 10.1007/978-3-642-38348-9_19
[17] *Bertoni G., Daemen J., Peeters M., Van Assche G.* Sufficient conditions for sound tree and sequential hashing modes // International Journal of Information Security. 2014. Vol. 13, issue 4. Pp. 335-353. DOI: 10.1007/s10207-013-0220-y
[18] *Гапанович Д.А., Чубариков В.Н.* Хэш-алгоритм с управляющей древовидной структурой и метод его реализации на параллельных архитектурах // Современные информационные технологии и ИТ-образование. 2017. Т. 13, № 1. С. 35-42. DOI: 10.25559/SITITO.2017.1.489
[19] *Минеев М.П., Чубариков:В.Н.* Лекции по арифметическим вопросам криптографии. НИЦ «Луч», 2014. 224 с.
[20] *Chapweske J., Mohr G.* Tree Hash EXchange format (THEX), 2003. URL: http://adc.sourceforge.net/draft-jchapweske-thex-02.html (дата обращения: 11.04.2018).
[21] *Dodis Y., Reyzin L., Rivest R., Shen E.* Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6 / O. Dunkelman ed. // Fast So ware Encryption. Lecture Notes in Computer Science. Vol. 5665, Springer-Verlag Berlin Heidelberg, 2009. Pp. 104-121. DOI: 10.1007/978-3-642-03317-9
[22] *Sarkar P., Schellenberg P.J.* A parallelizable design principle for cryptographic hash functions [Электронный ресурс] // Cryptology ePrint Archive, Report 2002/031, 2002. URL: http://eprint.iacr.org (дата обращения: 11.04.2018).
[23] *Bellare M., Guerin R., Rogaway P.* XOR MACs: new methods for message authentication using finite pseudorandom functions / D. Coppersmith ed. // Advances in Cryptology - CRYPTO '95. Proceedings of the 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27–31, 1995. Springer-Verlag Berlin Heidelberg, 1995. Pp. 15-28. DOI: 10.1007/3-540-44750-4
[24] *Bellare M., Micciancio D.* A New Paradigm for Collision-free Hashing: Incrementality at Reduced Cost / W. Fumy // Advances in Cryptology — EUROCRYPT '97. Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, May 11–15, 1997. Springer, Berlin, Heidelberg, 1997. Pp. 163-192. DOI: 10.1007/3-540-44750-4
[25] *Barreto P., Rijmen V.* The Whirlpool Hashing Function. NESSIE Report, Revised, May 2003.
[26] *Damgard I.B.* A design principle for hash functions / G. Brassard ed. // Advances in Cryptology - Crypto '89. Conference proceedings. LNCS, Vol. 435, Springer, New York, NY, 1989. Pp. 416-427. DOI: 10.1007/0-387-34805-0
[27] *Maurer U., Renner R., Holenstein C.* Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology / M. Naor ed. // Theory of Cryptography. TCC 2004. Lecture Notes in Computer Science. Vol. 2951, Springer, Berlin, Heidelberg, 2004. Pp. 21-39. DOI: 10.1007/978-3-540-24638-1_2

Поступила 11.04.2018; принята в печать 25.05.2018;
опубликована онлайн 30.06.2018.

References

[1] Kazymyrov A., Kazymyrova V. Algebraic Aspects of the Russian Hash Standard GOST R 34.11-2012, 2013. 18 p. Available at: http://eprint.iacr.org/2013/556.pdf (accessed 11.04.2018).
[2] Dourado E., Brito J. Cryptocurrency. *The New Palgrave Dictionary of Economics*, Online Edition, 2014. 10 p. Available at: https://www.mercatus.org/system/files/cryptocurrency-article.pdf (accessed 11.04.2018).
[3] Reshevsky M. Gold fever XXI century. *Computer Bild*. 2011; 17:64-69. (In Russian)
[4] Ammous S. The Bitcoin Standard: The Decentralized Alternative to Central Banking. New York: John Wiley and Sons Inc, 2018. Pp. 304.
[5] Aumasson J.-P., Neves S., Wilcox-O'Hearn Z., Winnerlein C. BLAKE2 – fast secure hashing, 2013. Available at: https://blake2.net (accessed 11.04.2018).

[6] Bertoni G., Daemen J., Peeters M., Van Assche G. Sakura: a flexible coding for tree hashing. Cryptology ePrint Archive. Report 2013/231, 2013. Available at: http://eprint.iacr.org (accessed 11.04.2018).

[7] Ferguson N., Lucks S., Schneier B., Whiting D., Bellare M., Kohno T., Callas J. The Skein Hash Function Family. Schneier on Security. Version 1.3. – 1 October, 2010. 92 p. Available at: https://www.schneier.com/academic/skein (accessed 11.04.2018).

[8] Kaminsky A., Radziszowski S.P. A case for a parallelizable hash. *Proceedings of 2008 IEEE Military Communications Conference MILCOM 2008*, San Diego, CA, 2008. Pp. 1-7. DOI: 10.1109/MILCOM.2008.4753182

[9] Rivest R.L., Agre B., Bailey D.V., Crutchfield C., Dodis Y., Fleming K.E., Khan A., Krishnamurthy J., Lin Y., Reyzin L., Shen E., Sukha J., Sutherland D., Tromer E., Yin Y.L. The MD6 hash function – a proposal to NIST for SHA-3, Submission to NIST, October, 2008. Available at: http://groups.csail.mit.edu/cis/md6 (accessed 11.04.2018).

[10] Dali F.A., Mironkin V.O. Review of approaches to the construction of tree-like modes of operation of some hash functions. *Information Security Problems. Computer Systems*. 2017; 2:46-55. (In Russian)

[11] Dali F.A., Mironkin V.O. On some models of tree hashing. *Obozrenie prikladnoi i promyshlennoi matematiki* = Applied and Industrial Mathematics Review. 2017; 24(4):241-244. (In Russian)

[12] Dali F.A., Mironkin V.O. On the tree modes of hash functions. *Information Security Problems. Computer Systems*. 2018; 1:113-121. (In Russian)

[13] Merkle R.C. Secrecy, authentication, and public key systems. PhD thesis, UMI Research Press, 1982. 104 p.

[14] Dali F.A., Mironkin V.O. On probabilistic characteristics of a single class of tree hashing models. *Obozrenie prikladnoi i promyshlennoi matematiki* = Applied and Industrial Mathematics Review. 2016; 23(4):345-347. (In Russian)

[15] Mironkin V.O. Urusov M.I. On collisions of Merkle trees. *Obozrenie prikladnoi i promyshlennoi matematiki* = Applied and Industrial Mathematics Review. 2018; 25(1):84-88. (In Russian)

[16] Bertoni G., Daemen J., Peeters M., Van Assche G. Keccak. Johansson T., Nguyen P.Q. (eds). *Advances in Cryptology – Eurocrypt 2013. Eurocrypt 2013*. Lecture Notes in Computer Science. Vol. 7881. Springer, Berlin, Heidelberg, 2013. DOI: 10.1007/978-3-642-38348-9_19

[17] Bertoni G., Daemen J., Peeters M., Van Assche G. Sufficient conditions for sound tree and sequential hashing modes. *International Journal of Information Security*. 2014; 13(4):335-353. DOI: 10.1007/s10207-013-0220-y

[18] Gapanovich D.A., Chubarikov V.N. Hash algorithm with the controlling tree-like structure and the method of its implementation on parallel architectures. *Modern Information Technology and IT-education*. 2017; 13(1):35-42. DOI: 10.25559/SITITO.2017.1.489 (In Russian)

[19] Mineev M.P., Chubarikov V.N. Lectures on arithmetic questions of cryptography. Moscow: Scientific and Publishing Center «Ray», 2014. 224 p. (In Russian)

[20] Chapweske J., Mohr G. Tree Hash EXchange format (THEX), 2003. Available at: http://adc.sourceforge.net/draft-jchapweske-thex-02.html (accessed 11.04.2018).

[21] Dodis Y., Reyzin L., Rivest R., Shen E. Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6. O. Dunkelman ed. *Fast So ware Encryption*. Lecture Notes in Computer Science. Vol. 5665, Springer-Verlag Berlin Heidelberg, 2009. Pp. 104-121. DOI: 10.1007/978-3-642-03317-9

[22] Sarkar P., Schellenberg P.J. A parallelizable design principle for cryptographic hash functions. Cryptology ePrint Archive, Report 2002/031, 2002. Available at: http://eprint.iacr.org (accessed 11.04.2018).

[23] Bellare M., Guerin R., Rogaway P. XOR MACs: new methods for message authentication using finite pseudorandom functions. D. Coppersmith ed. *Advances in Cryptology - CRYPTO '95. Proceedings of the 15th Annual International Cryptology Conference*, Santa Barbara, California, USA, August 27–31, 1995. Springer-Verlag Berlin Heidelberg, 1995. Pp. 15-28. DOI: 10.1007/3-540-44750-4

[24] Bellare M., Micciancio D. A New Paradigm for Collision-free Hashing: Incrementality at Reduced Cost. W. Fumy. *Advances in Cryptology — EUROCRYPT '97. Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques Konstanz*, Germany, May 11–15, 1997. Springer, Berlin, Heidelberg, 1997. Pp. 163-192. DOI: 10.1007/3-540-44750-4

[25] Barreto P., Rijmen V. The Whirlpool Hashing Function. NESSIE Report, Revised, May 2003.

[26] Damgard I.B. A design principle for hash functions. G. Brassard ed. *Advances in Cryptology - Crypto '89. Conference proceedings.* LNCS, Vol. 435, Springer, New York, NY, 1989. Pp. 416-427. DOI: 10.1007/0-387-34805-0

[27] Maurer U., Renner R., Holenstein C. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. M. Naor ed. *Theory of Cryptography. TCC 2004.* Lecture Notes in Computer Science. Vol. 2951, Springer, Berlin, Heidelberg, 2004. Pp. 21-39. DOI: 10.1007/978-3-540-24638-1_2

Submitted 11.04.2018; revised 25.05.2018;
published online 30.06.2018.

About the authors:

Dmitriy S. Bogdanov, Senior researcher, Laboratory of TVP (47 Nakhimovsky Avenue, Moscow 117418, Russia), ORCID: http://orcid.org/0000-0001-6178-6420, bogdanovds@rambler.ru

Farkhad A. Dali, expert, Technical Committee of Standardization Cryptography Information Security (Building 1, 1/23 Old Petrovsky-Razumovsky passage, Moscow 127287, Russia), ORCID: http://orcid.org/0000-0002-3344-9766, dali_fa@tc26.ru

Vladimir O. Mironkin, Senior lecturer, Department of Computer security, National Research University Higher School of Economics (34 Tallinn Str., Moscow 123458, Russia), ORCID: http://orcid.org/0000-0002-5606-7509, mironkin.v@mail.ru